A Just-In-Time Information Retrieval Agent that Provides Context Aware Support of Text Creation

Elitza Chenkova and Ivan Koychev[1]

Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski"
elichenkova@gmail.com, koychev@fmi.uni-sofia.bg

Abstract. Finding relevant resources and supplemental information that support the process of text creation is time consuming and often distracting for the author. With the increase of digitally stored information, accessing diverse information resources is easier but at the same time it gets harder and harder to find the relevant information giving you full view on the subject you write or read about. We propose a solution for getting the needed contextual view over the relevant accessible resources - an autonomous just-in-time information retrieval agent for automatic context information finding. The agent extracts keywords, searches and suggests relevant resources. Having the process of gathering relevant information automated greatly improves the user's productivity and saves up her/his time.

Introduction

Imagine you are a corporate low firm employee and has been given the task to prepare the contract for merging too big companies. Now, before sitting down to prepare the contract, first you need to consult the law. Afterwards, it would be nice to see some other similar merge-arranging documents. Furthermore, you may need to consult some different papers stating your clients' preferences and needs. These preparatory actions take time and make you go deep into the firm archives and digital resources, where you can easily get lost if you don't have the experience to deal with vast amounts of information. Wouldn't it be so much nicer if someone can do all the papers digging for you and present you with a nicely ordered folder containing everything you need to know to prepare that contract?

Now imagine you are a news paper column writer. Recently an influential figure in the public life has made a statement you want to comment about. You open your text editor and prepare to start writing. But just then you figure out you don't remember some details about the event, or about previous actions on behalf of this person, or may be comments from other important people on the subject. You need to go through the digital archives of yours and other journals, and also may be some other media's publications. Wouldn't you then prefer to have all this information easily accessible through your text editor interface giving you even suggestions on the most relevant publications?

Both examples show how our work is dependent on knowing the proper relevant information. In order to achieve this we spend hours informing ourselves. In this paper we present our research done on the automation of this process. We develop an information retrieving intelligent agent, which autonomously

[1] Also associated with Institute of Mathematics and Informatics - BAS.

and proactively extracts keywords from the current paragraph of the document that is being created or just read at the moment. The agent uses these keywords to perform search for relevant documents through the resources available to the user.

To conduct our research we focused on the following tasks:

- Examine thoroughly existing methods and algorithms for keywords extraction and relevant documents retrieval and choose the best suitable techniques for our case (Section 3).
- Design an agent for contextual search – determine its architecture and functionality requirements. These are presented in Section 4.
- Implement a prototype of the agent as a MS Word 2007 Add-in for automatic search of context information, using integration with Windows Desktop Search. Brief overview of the implementation details is given in Section 5.
- Conduct initial user tests on the implemented prototype in order to gather feedback from users. The test results are mentioned in Section 6.

But first, before focusing on these tasks, we explain in more details the problem and the chosen solution in Section 2.

Problem and Proposed Solution

Whenever we need to compose any business related document, we depend on information. It doesn't matter if we have to write a corporate email or a review on a book. We need to consult the context related resources. These might be the things that recently happened in the company, any major events concerning the topic, or we might want to know other critics' opinions on that book. Finding this information is not that trivial - it needs planning and it takes time. You have to know exactly what you are looking for and where you can search for it. With the birth of digital data storing, accessing information and keeping big data archives became much easier, and people were greatly helped on their quest for knowledge. We now have search engines, which we can ask to find related resources on the web about anything we want to know. We have big corporate archives where we can search for any document that we might ever need. We have digital libraries giving us access to thousands of papers. Technologies can make finding relevant information on time come true.

However, being more efficient and saving more time is the goal of any company. Your boss would always want you to have finished that report earlier. At the same time, you would prefer to just focus on the report and not spend your time gathering the documents you need to review. On the other hand, when reading your work, your boss would prefer to have these documents at hand too, as well as other data on the subject available only to her/him perhaps.

Being able to focus on writing or reading tasks without the distraction of searching for relevant information is the problem that we focus on. The solution we propose is a Just-In-Time Information Retrieval (JITIR) [1] agent which will autonomously determine the user's current context by extracting keywords from the text she/he is working on and search for relevant accessible resources based on these keywords. The main argument for using just-in-time autonomous

information retrieval is that users of such agents are not merely more efficient at retrieving information, but actually retrieve and use more information than they would with traditional search engines.

A JITIR agent has three main features: *proactivness, a nonintrusive yet accessible interface, and attention to local context*. Having keywords extracted from the document and even the paragraph that is currently worked on gives nice focus on the user's context. This is done autonomously; the user doesn't have to spare time or attention on this process. Then a search is conducted throughout resources available to the user and a list of relevant documents is prepared. This list is then presented to the user in a nonintrusive manner so that any distraction can be avoided and the user can keep on with the work as long as she/he doesn't need to pay attention to the suggested resources. In this way, with the help of such an agent, the user won't waste time or lose focus searching for relevant information while working on a document and therefore keep up and even increase her/his efficiency because of the easier access to information.

Chosen Techniques

In this section we briefly describe the solutions we chose for the two main tasks of our JITIR agent – keywords extraction and retrieval of relevant documents. Before being able to determine anything about the importance of a word, there are certain pre-processing steps that need to be performed to facilitate the keywords extrction.

Text Pre-processing

First of all, in any text there are words which on their own do not bare any meaning. These are conjunctions, prepositions, etc. They are called 'stop words' and are being filtered out of the text by the use of 'stop words' lists. We used one list for the Bulgarian and one for the English language. Furthermore, words need to be normalized so that if just written differently, they are not counted as different words. For example, the normalized version of 'USA' and 'U.S.A.' would be just 'usa'. Finally, words should be reduced to a common base form. That is, if we have 'running', 'runners' and 'runs', they all need to be reduced to just 'run'. To do this, one can use stemming techniques where the ends of words are chopped off based on heuristics and the goal is achieved correctly most of the time; or use lemmatization, which does this with the use of a dictionary and morphological analysis of words, aiming to return the base or dictionary form of a word. Although lemmatization normally gives more proper results, for the sake of computational effectiveness we chose to use a stemmer. For English, we used the Porter's stemming algorithm [2] – the most commonly used one, which has proved to be fairly effective and for Bulgarian we used a stemmer, designed by Preslav Nakov - BULSTEM [3].

Keywords Extraction

After studying different keywords extraction techniques, we decided to use statistical methods to determine how representative words are for the observed

document. First we will describe the details of the chosen technique and then give the reasons for our choice.

As soon as a document text has been pre-processed, we can start estimate how important its words are. To do this, we use simple, yet powerful measure: Relative Frequency Ratio (RFR) [4]. It measures the weight of a term as the ration of its frequency in the current document towards its frequency in a general collection (corpus) of documents:

$$RTF_{t,d} = \frac{TF_{t,d}}{CF_t} \quad (1)$$

t – the term we want to measure

d – the document, for which we measure the term

$TF_{t,d}$ - Term Frequency – the frequency of the term t in the document (how many times this term occurs in the document).

CF_t - Collection Frequency – the frequency of the term t in the whole corpus (how many times this term occurs in the corpus).

The idea is that words that occur relatively frequently in a certain text compared to how frequently they occur in a general text are more likely to be good keywords. In our case we want to determine keywords per active paragraph, based on the word distribution in the entire document. Therefore, the entire document is our corpus, and the paragraphs are its entities. According to this definition, we calculate the RFR of each word in each paragraph and determine the keywords per paragraph.

The main reason for choosing a statistical measure is again efficiency. Keywords have to be re-determined every time the user changes the text, therefore calculations needs to be as light and simple as possible.

Relevant Documents Retrieval

After determining the keywords per active paragraph, we construct a search phrase out of the two words with the highest score and find documents relevant to this phrase. Retrieving documents is based on determining their relevance to certain search phrase. This can be done using different models of the documents collection, but for our problem we decided to use an already developed Desktop Search system. The reason for choosing this solution instead of implementing a search engine on our own is the fact that there has been a lot of development on the topic lately. This is due mainly to the constantly increasing size of hard disks nowadays, as well as the observation that more than 80% of the information resources of a company are kept in an unstructured manner – all over the employees' hard disks, in files and directories, emails, etc. The business needs an easy way to access and search these resources and the industry is racing to produce it.

Since we decided to rely on an outsource system for the work of our agent, we had to determine which one has the highest probability to be present on a future user's desktop. After going through some comparative reviews over the most popular desktop search tools, we decided to use the Windows Desktop Search system given its good feedback, high popularity and relatively easy integration.

Architecture and Functionality Requirements

The main functionality requirements towards our agent in order to solve the problem of providing context relevant information are:

1. Constantly monitor the user text environment and synthesize keywords information in search phrases.
2. Find relative resources based on the current search phrase sending requests to the Desktop search tool.
3. Display the results in a non intrusive manner through easily accessible and light sideway interface.

These functionalities are implemented in the architecture design showed in Fig. 1.

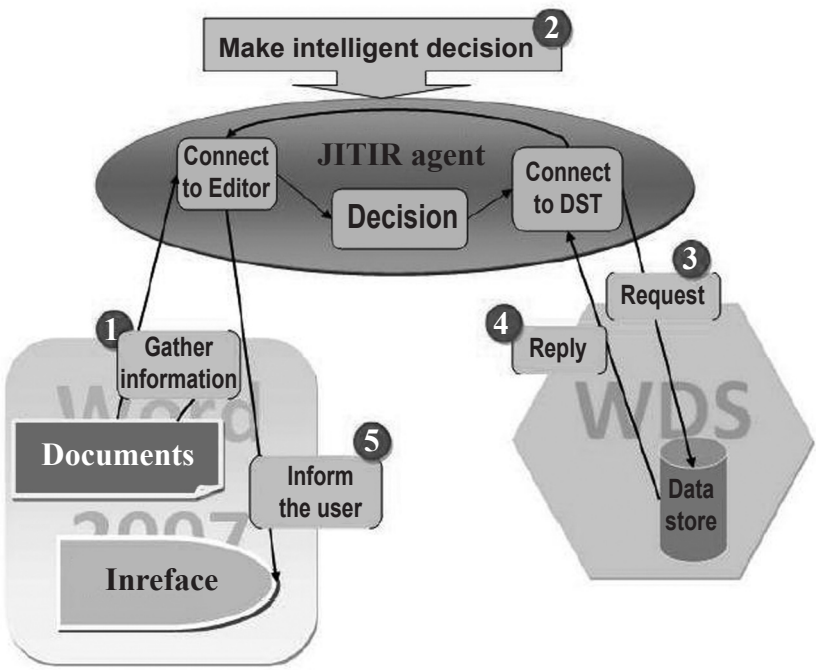

Fig. 1. Architecture of the agent.

Every couple of minutes the agent is checking the text editor environment for change in the text or change in the currently active paragraph. Whenever there is change in the text, the document is re-indexed and words RFR weights are recalculated. Based on the active paragraph keywords are selected and a search phrase request is constructed. This request is forwarded to the Desktop Search tool (DST). The reply is processed and the most relevant resources are displayed in the agent interface.

Implementation

A prototype of the agent was implemented as an add-in for the Microsoft Office text editor Word. Recently Microsoft lounged technology allowing the development of all kinds of add-in applications for their Office products, so implementing this agent gave us nice opportunity to test this technology. The add-in was developed for Microsoft Office Word 2007. For the desktop search functionality, we integrated our agent with Windows Desktop Search – the Microsoft tool for searching Windows desktops. It has good interaction with the file system and allows for easy integration. It also gives possibilities for using search filters, so that search can be conducted on emails, documents or only program files, for example.

The add-in was developed in C# on Microsoft Visual Studio .NET 2005 v.8.0 Professional Edition with Visual Studio Tools for Office 2005 SE (VSTO) installed. The implementation took longer than expected mainly due to lack of documentation of the still new technology at that time.

User Interface

After installation, the agent can be found in the Add-Ins tab of MS Office Word 2007. Its interface is fitted in a vertical panel, which is shown and respectively hidden by clicking the Context Information button from the ribbon in the Add-Ins tab. The agent's default behaviour is to act autonomously and constantly index the text in the current document and enquire relevant documents. This behaviour is indicated by a check in the Auto-Refresh checkbox. Whenever the user wants to use the agent manually, the checkbox has to be unchecked and then, by clicking the Index button the text is re-indexed and the newly extracted keywords are shown in the search box. Relevant results among documents on the Desktop are shown automatically in the list below together with their ranking, according to the Windows Desktop Search engine. Each line of the results list is clickable, opening the corresponding document in the associated program. The default WDS filters are shown in a drop-down list below the Search box, allowing the user to select where she/he wants the search to be performed – only in documents or emails, etc. When the Auto-Refresh box is unchecked, the user can also enter terms in the Search box manually and perform queries by clicking either Search the Desktop or Search the Web buttons. Search the Web button is active also in the autonomous behaviour state, since relevant web resources are not obtained automatically and this button has to be used explicitly if the user wants to search the web for related documents. Clicking the button opens the default web browser of the user and sends request to the Google search engine with the query terms from the Search box.

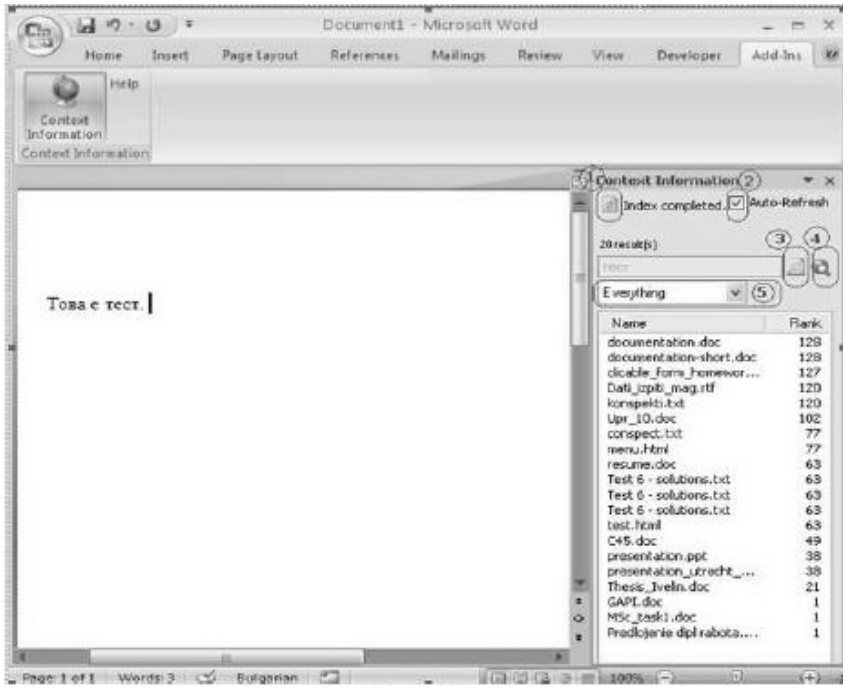

Fig. 2. User Interface: 1.Indexing button. 2. Change behaviour button. 3. Search the Desktop button. 4. Search the Web button. 5. Filters list.

User Feedback

Initial user tests were conducted over several MS Office Word users, which gave overall positive feedback on the add-in. Their main criticism was on performance issues, which can hopefully be overcome with better understanding of the VSTO technologies. Also interesting suggestions for more features came through feedback, like having the feature to automatically search not only desktops but also the Web.

Related Work

There has been developed a couple of tools that try to give contextual information to text readers and writers. Two of them are described in more details in [1] - the Remembrance Agent and Margin Notes. Remembrance Agent is the closest one to what we developed - it suggests personal emails and documents based on text being written or read in the UNIX text editor Emacs. Margin Notes assists only Web readers by automatically annotating Web pages as they are being loaded into a browser, adding hyperlinks to personal files. It compares each section of the loaded document to preindexed email archives, notes files, and other text files, based on keyword cooccurrence. Another JITIR agent is Lumiere [5], which was used as the basis for the Microsoft Office Assistant. It tries to infer users' needs by considering their background, actions, and queries, employing Bayesian user models.

Conclusions

This paper presents our research on supporting text editor users by proposing contextually relevant resources. We develop a JITIR agent which constantly monitors the text modified or read by the user, determines context by keywords extraction and searches for available to the user relevant resources by phrases constructed from keywords. The agent was implemented as an MS Office Word 2007 add-in application prototype for English and Bulgarian texts, working in integration with the MS Windows Desktop Search tool.

The feedback received on the prototype gave us some directions for future work. With professional help from VSTO developers and cognitive specialists we can make a more robust and user friendly tool. Another track of implementation research is developing an agent prototype for Linux users, which will increase the range of people using it and respectively amount of feedback and quality of the product.

Acknowledgements. This research was supported by the Smart Book project, subsidized by the Bulgarian National Science Fund, under Grant D002-111 /15.12.2008.

References

1. Rhodes, B.J., Maes, P.: Just-in-time information retrieval agents. IBM Systems Journal 39(3-4), 685-704 (2000)
2. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130-137 (1980)
3. Nakov, P.: BulStem: Design and Evaluation of Inflectional Stemmer for Bulgarian. In: Proceedings of Workshop on Balkan Language Resources and Tools, Thessaloniki, Greece (2003)
4. Damerau, F.J.: Generating and evaluating domain oriented multi-word terms from texts. In: Information Processing and Management: an International Journal 29, 433-447 (1993)
5. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K.: The Lumiere project: Bayesian user modelling for inferring the goals and needs of software users. In: Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference, 256-265 (1998)